# A Computational Framework for Vertical Video Editing

Vineet Gandhi, Rémi Ronfard

Eurographics Workshop on Intelligent Cinematography and Editing (2015), pp. 1–7
W. Bares, M. Christie, and R. Ronfard (Editors)

# A Computational Framework for Vertical Video Editing

Vineet Gandhi and Rémi Ronfard

Inria, Univ. Grenoble Alpes, LJK, France

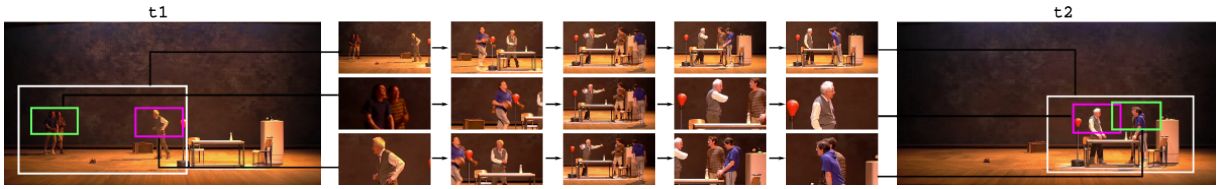


**Figure 1:** *Vertical editing in action. Our system proposes sub-frames at times $t_1$ and $t_2$ and automatically computes virtual pan, tilt and zoom camera movements that interpolate between the editor's choices. In this example, we are showing 3 of 9 possible shots. The number of possibilities increases with the number of actors on-screen. Because they are computed off-line, the generated shots anticipate the actor's movements and keep the composition balanced, making them easier to edit into a complete movie.*

**Abstract**

*Vertical video editing is the process of digitally editing the image within the frame as opposed to horizontal video editing, which arranges the shots along a timeline. Vertical editing can be a time-consuming and error-prone process when using manual key-framing and simple interpolation. In this paper, we present a general framework for automatically computing a variety of cinematically plausible shots from a single input video suitable to the special case of live performances. Drawing on working practices in traditional cinematography, the system acts as a virtual camera assistant to the film editor, who can call novel shots in the edit room with a combination of high-level instructions and manually selected keyframes.*

Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Vision and Scene Understanding]: Video analysis—I.3.3 [Computer Graphics]: Picture/Image Generation—

## 1. Introduction

*Vertical editing* is a term coined by film editor Walter Murch to describe editing of the image within the frame, as opposed to arranging shots along a timeline, which is considered horizontal editing.

> *Up until now, motion picture editors have thought almost exclusively in the horizontal direction. The question to be answered was simply, "What's next?" - that' s complicated enough - there are a tremendous number of options in the construction of a film. In the future, that number is going to become even more cosmic because film editors will have to start thinking vertically as well, which is to say: "What can I edit within the frame?" [Mur01]*

In this paper, we propose a computational framework for helping the editor with the cosmic choice of vertical editing in the restricted case of stage perfomances, by leveraging recent work in computer vision. In traditional filmmaking, the choice of framing is considered to be an essential element of cinematography, and is controlled by a dedicated camera assistant, with a good knowledge of the rules of cinematography, including frame composition and timing [TB09]. The same rules are applicable to the case of vertical editing in post-production. Reframing can be performed manually by creating keyframes at discrete time in-

stants, and interpolating the keyframes to produce a smooth trajectory of the frame. Modern film editing suites such as Avid Media Composer, Adobe Premiere or Final Cut Pro offer support for such key-framing techniques under the name of 'Ken Burns' effect [†]. Because the keyframes are linearly interpolated, the approach typically requires a large number of keyframes, painstakingly drawn through trial-and-error. To remedy that situation, we propose a novel content-aware interpolation method which guarantees that the framing remains cinematically valid, at least in a simplified computational model of cinematography.

We focus on the important case of live performances recorded from a small number of fixed viewpoints, including live concerts, opera and theatre performances. This has important application for preserving cultural heritage [McA09] where the dilemma is to choose between a single-viewpoint camera covering the entire stage, which is cost-efficient but not attractive to viewers, and a multi-camera setup, whose cost may exceed the budget of the original production. Combined with very high resolution video (4K), the techniques described in this paper offer a compromise which can be both cost-efficient and high quality. One further benefit of our approach is that we can generate content for multiple formats, i.e. cinema, television, tablets and even smartphones. Contrary to the case of retargeting, we are not betraying the film director's intentions but instead translating the stage director's production into different screen idioms, i.e. long shots for the cinema, medium shots for television and close shots for tablet and smartphones.

Section 2 present a review of related work. Section 3 discusses the topic of frame composition and presents our taxonomy of shots. Section 4 describes the core of our method, which consists in three main steps - tracking actors, computing frame constraints and optimization. We present experimental results in Section 5 on three different tasks - reframing short sequences to static shots, reframing short sequences to dynamic shots using keyframes, and reframing long sequences with multiple keyframes.

## 2. Related Work

Editing of live action video is a challenging area of research [KAAU02, HLZ04] which must face the difficult problem of detecting, localizing and recognizing the important elements of the scene. As a result, previous work has been limited to restricted situations such as lectures and conferences [Bia04, HWG07, ZRCH08], usually with only one speaker and a restricted course of events. Recent work has looked at the important problem of *when* to place cuts and transitions during video editing [BLA12], assuming a given set of shots. In contrast, we are dealing here with the complementary problem of generating those shots in the first place.

Computational support for vertical editing has already been proposed for real-time broadcasting of sports events [KWK12, SFWS13] using a variety of visual motion cues which can be computed efficiently. Our contribution instead focuses on providing offline tools for post-production suitable to the case of contents with high cultural and historical values, where offline actor detection and naming methods can be used to provide content-aware interpolation, although at a much higher computational cost.

A more general approach to the vertical editing problem is to predict where the audience's attention focuses in the original frame and to automatically reframe according to the prediction [GWP07, CLM08]. This includes work by Deselaers et al. on automatic pan and scan [DDN08] and work by Jain et al. on video re-editing, where automatic algorithms are proposed for simultaneously reframing and re-editing the video [JSSH14]. In contract to this previous work, our system is interactive, letting the editors define their choices of framings at different keyframes and offering content-aware interpolation between selected keyframes.

Our work is also related to recent video stabilization methods [GKE11, LGW*11], where the original video is rendered as if it has been recorded by a smooth camera path, by designating a virtual crop window of predefined scale. These video stabilization algorithms are based on keypoint feature tracking and camera motion estimation in the form of 2D transformation. Instead of keypoint tracks we use recently proposed human detection methods [GR13, DTS06, FGMR10], to define boundary constraints. The goal is then to find a smooth cropping window path (virtual pan-tilt-zoom) best representing the chosen type of shot and adhering to the constraints and the aspect ratio. In recent work, we proposed an algorithm for automatically generating a small number of cinematographic rushes along those lines [GRG14]. Those automatically generated rushes can then be edited together manually. One unresolved problem with that approach is how to choose which rushes to generate in the first place. In this communication - which is based on the same computational framework - we pursue another route, which is to let the film editor make vertical choices (framing) and horizontal choices (cutting) interactively by defining a small number of keyframes and interpolating between the keyframes. We refer the reader to [Gan14] for a comparison of the two approaches.

## 3. Grammar of the shot

For our purpose, a shot is a temporal sequence of frame compositions (framings), The framings in a shot can change due to movement of the virtual camera, movement of the actors or both. In this section, we introduce a taxonomy of framings and shots which is described in details elsewhere [RGB13]

[†] Ken Burns is an American documentary film-maker who made frequent use of pan-and-scan techniques over still photographs.

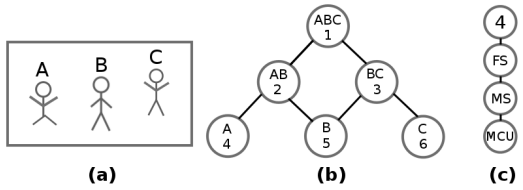


**Figure 2:** *(a) A given scene with three actors A, B and C. (b) The shot graph for that scene. Each node in the graph represents a shot with a constant framing. Each edge in the graph represents a shot with a transition between different framings. Framings within each framing category can vary in size. The tightest framing is bounded by the boundaries of actors contained within the subframe. The widest framing is bounded by the boundaries of the actors outside the subframe. (c) Each node is then further subdivided into different shot sizes.*

and explain how the sub-shots extracted from a given video can be organized into a shot graph.

### 3.1. Frame composition

We use a taxonomy of framing categories which allows us to partition the space of framings into well-defined classes. We use the simplest possible taxonomy, where a framing is identified with the list of actors present in the shot from left-to-right and their size on the screen. We use the classification of shot sizes adopted by film scholar Barry Salt [Sal06]. For the purpose of clarity and exposition, we use only thee shot size categories - full shot, medium shot and medium close-up. In a scene with $N$ actors, a maximum number $T * N(N+1)/2$ of sub-framings can be defined, where $T$ is the total type of shot sizes being considered. Note that the sub-framing can be described with just the names of the left-most and right-most actors present in the shot.

### 3.2. Simple and complex shots

Thomson [TB09] distinguishes three types of shots, based on which of the following four items are moving - the subjects, the lens, the pan and tilt head, and the camera mounting. A simple shot is composed of a subject made without movement of the lens, pan and tilt head or the camera mounting. A complex shot is composed of a moving subject made with movement of either the lens or the pan-tilt head, or both. The developing shot is composed of a subject with movement of the camera mounting. In our case, the input video is a simple shot, and the reframed shots are either simple shots (if we keep the frame window constant) or complex shots (if we move the frame window). Developing shots are left for future work.

### 3.3. Shot graph

In the taxonomy of framings introduced in the last section, the set of possible sub-framings can be represented in a shot graph, as illustrated in Fig 2. Each node in this graph represents the set of possible framings which lies within the tightest and the most relaxed framing for the given set of actors. The topmost or the parent node represents the sub-framings containing all the actors in the scene and the leaf nodes represents the framings containing an individual actor. Each edge in the shot graph represents a possible transition between framings.

Transitions between framings can be caused by lens movements. For example, zooming-in can push an actor out of the frame. Zooming out can pull an actor into the frame. Transitions can also be caused by pan-and-tilt movements. For example, a left-to-right pan shot will reveal actors entering the frame from the right and hide actors exiting the frame on the left. We combine the effects of the lens and the pan-tilt head into camera edges.

Finally, transitions between framings can be caused by actor movement even with a fixed camera. Typical events are actors entering and exiting the frame, changing positions within the frame, hiding each other or showing up from hiding.

At any given time, a framing can have at most four edges, corresponding to the following events

1. The left-most actor inside the frame moves out of the frame (arrow down and right in the shot graph)
2. The right-most actor indside the frame moves out of the frame (arrow down and left in the shot graph)
3. The left-most actor outside the frame moves into the frame (arrow up and left in the shot graph)
4. The right-most actor outside the frame moves into the frame (arrow up and right in the shot graph)

The same four transitions between framings can also be caused by actor movements. In that case, the naming of the shot changes, but the camera frame does not move.

## 4. Content-aware interpolation

In this section, we describe how we precompute the shot graph for an entire video and use it to generate arbitrarily complex shots at runtime. As a first step we perform actor detections on given video sequence at discrete intervals and then combine them into actor tracks using an interpolation method. This is described in subsection 4.1. Using these tracks we can have the information about the actors present at any given frame in the video sequence with their corresponding positions in image coordinates and a relative depth coordinate which can be approximated using the size of the detections windows. We further take advantage of a fixed stage to estimate the floor projections of the detection windows. This floor plan view can be then used to compute

the boundary constraints at any given frame of the video, and then to build a shot graph representing all possible sub-framings or compositions for a given frame, as described in Fig. 2.

The end goal is to extend these sub-framings into temporal shots. This is done using the convex optimization method proposed by Gandhi et al. [GRG14, Gan14], which takes as input the boundary constraints and the actor detections, and computes a smooth interpolating window best representing the true actor motion on stage, while adhering to the given boundary constraints.

### 4.1. Tracking of actors

We use a generative model for learning person and costume specific detectors based on a stable color region descriptor which has been shown to be robust to large appearance variations in image retrieval [SZ03] and movie indexing tasks [GR13]. Following the approach proposed by Gandhi and Ronfard [GR13], we learn each actor model from a small number of labeled keyframes or video tracks. Ideally a sequence of each actor performing a 360 degree turn would be sufficient to build such models, this can be done conveniently for live performances. More specifically, the actor's head and shoulders are each represented as a constellation of optional color regions and the detection can proceed despite changes in view-point and partial occlusions. These multi-view models are then used in a maximum likelihood framework for actor detection. The final result are upper body detections with the actor labels as shown in Fig.3. Using upper body detectors gives an added advantage of pose invariance and the ability to detect sitting, standing or partially leaning actors in contrast to commonly used upright pedestrian detections.

In addition to tracking and recognizing actors, we compute their projections on the stage floor. This information is necessary for defining the frame compositions in arbitrary poses like sitting, lying down etc. To calculate stage floor projections, we first approximate the stage floor by the equation of a plane in the $3D$ world coordinates using three or more manually labelled interest points. Now the actor's upper body can be placed in this $3D$ coordinate system using the image coordinates of the upper body detection center $(x_i, y_i)$ and the size of the detection window as the depth coordinate.


**Figure 3:** *Example detections with projections drawn on the stage floor.*

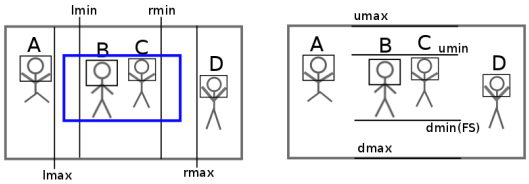


**Figure 4:** *This figure illustrates the frame boundaries for containing Actors (B,C) in a Full Shot (FS). The horizontal constraints (lmin, rmin, lmax, rmax) and vertical constraints(umin, umax, dmin, dmax) are computed using the coordinates of the actor detection windows. The value (umin,dmin) depends on the type of shot being considered, shown is an example with a full shot.*

### 4.2. Boundary constraints

Given the actor detections, we can now compute boundary constraints for the given set of actors in all frames. This is illustrated in Fig.4. The $(lmin, rmin, lmax, rmax)$ are the horizontal constraints and can be derived using the coordinates of the upper body detections. Similarly, $(umin, dmin, umax, dmax)$ define the vertical constraints. For a composition larger then a full shot, we use floor point projections to compute the *dmin* values.

Constraints $(lmin, rmin, umin, rmin)$ define the tightest composition possible for a given set of actors and the constraints $(lmax, rmax, umax, rmax)$ define the widest composition possible for the given set of actors. The type of possible framings for a given set of actors are limited by these boundaries.

**Figure 5:** *All shot types can vary in sizes. Left: widest framings per shot type. Right: tightest framings per shot type. Top: four-shots. Second row : three-shots. Third row: two-shots. Bottom: one-shots.*

### 4.3. Reframing as optimization

Given some user-defined keyframes $g_k = (gx_k, gy_k, gs_k)$, we now use the boundary constraints $c_n = (lmin_n, rmin_n, lmax_n, rmax_n)$, to compute a dense and smooth framing $f_n = (fx_n, fy_n, fs_n)$ that approximates the $g_k$ and satisfies the constraints $c_n$. In previous work, we have shown how to cast such problems as a convex optimization problem with linear constraints [GRG14, Gan14], and we use that approach to compute a unique and exact solution.

## 5. Experimental results

We tested our algorithms on two hours of theatre performances, recorded in FullHD (1080p) during rehearsals using a Lumix GH2 with a 20 mm lens, directly facing the stage at a distance of 20 meters. The accompanying videos present examples of our results from three different plays to illustrate our results - A l'ouest, by Nathalie Fillion; Lorenzaccio, by Alfred de Musset ; and Death of a salesman, by Arthur Miller. We learned the actor models from separate training videos and computed the shot graphs using our prototype implementation, in MATLAB and OpenCV.

In each case, we split the recording into narrative segments (scenes and sub-scenes) of approximately one minute. We present three separate experiments for editing those segments. In the first experiment, we generate a simple (static) shot covering the entire segment. In the second experiment we generated a complex (dynamic) shot by choosing compositions at two different keyframes. In the third experiment, we generated a complex (dynamic) shot interpolating keyframes selected manually over the entire segment.

### 5.1. Simple shots

The goal here is to find the best static frame containing any given subset of the actors in a given type of composition for the given time interval, we call it a KEEP query. The user input is only the time interval and the desired composition. To solve this we first compute the constraints $c_n$ and rule based framing goals $g_k$ for the given set of actors and the type of shot at each instant of time. Then we solve for $fn$ where, $[\forall n : f_n = f = (fx, fy, fs)]$. While this mode is fully automatic, it can only be used in simpler cases, with few actors and little stage motion. In most situations, we need more input from the user, as in the next two experiments.

### 5.2. Complex shots with two keyframes

In this experiment, we let the user choose the framing at the beginning and the end of the sequence, and we compute a complex shot interpolating the keyframes. When the two keyframes have the same composition (same actors and sizes), we attempt to keep the same composition throughout

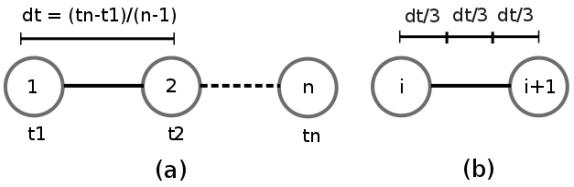


**Figure 6:** *Time partition for a (PAN FROM COMPOSITION TO COMPOSITION) complex shot with two keyframes. (a) After calculating the number of nodes to be traversed, the total time is first equally distributed for each edge. (b) The time allotted for each edge is then further divided into sections where we keep the node and where we perform the interpolation between them. Constraints are only defined for the segments where we keep a particular node.*

**Figure 7:** *Our method vs. linear interpolation. First row shows the starting and ending keyframes with the given bounding boxes. Second row shows the intermediate frame for a linear interpolation. Third row illustrates the corresponding intermediate frames using the proposed method.*

the sequence (we call this PAN WITH COMPOSITION). When the two keyframes have different compositions (different actors or different sizes), we instead search for a path in the shot graph leading from the first composition to the second composition. We explain the steps in each case.

**PAN WITH COMPOSITION.** Here the compositions is same throughout the sequence. The compositions at the first and the last keyframes are set as framing goals. The boundary constraints and the motion constraints for each actor are defined for the entire interval of time. The final interpolated shot is than calculated using the optimization function explained in Section 4.3. The first row in Fig.1 shows an example of this form of interpolation and the corresponding results are also shown in the attached video.

**PAN FROM COMPOSITION TO COMPOSITION.** Here the initial and the final composition are not the same. Both the initial and final compositions are approximated by the closest node in the shot graph and the shortest path between these nodes is calculated. The time partition is then calculated between these nodes as explained in Fig. 6. A framing goal is assigned for the composition at each intermediate node. The boundary constraints and motion constraints are defined, only for the time segments where a node is kept. The framing goals and the constraints are then used in the optimization framework to obtain the final interpolated shots. The second and third row in Fig. 1 show the examples of this form of interpolation.

Fig 7 compares the proposed method with linear interpolation, and illustrates how the proposed method method maintains better frame composition. In our experiments, we found that this strategy worked reasonably well on shorter segments with limited motion. On longer segments with more actor movements, additional user input is needed. This leads to our third experiment.

### 5.3. Complex shots with multiple keyframes

In our third experiment, we let user advance sequentially in a long video segment and pause at arbitrary keyframes. At a keyframe, the user can select any of the proposed framings in the shot graph. We can then compute the interpolating shot leading to that new keyframe, in time approximately equal to the segment duration. Thus, the user receives immediate feedback and can review his previous choices sequentially. We compute the interpolating segments by classifying intermediate shots between the previous and current keyframes as either a PAN FROM COMPOSITION TO COMPOSITION or a PAN WITH COMPOSITION, as explained in the previous section.

The resulting system is very easy to use even for novice users. Choosing the keyframes, selecting the framings, and reviewing the interpolating sub shots typically takes between 5 and 10 minutes per minute of video. We can therefore extrapolate than a full two-hour recording of a theatre performance can be vertically edited in two days. Future work is needed to verify this assumption.

## 6. Conclusion

We have presented tools for controlling a virtual pan-tilt-zoom camera during post-production using keyframe interpolation, using recent advances in computer vision for tracking and naming actors based on their visual appearance. Our main contribution is an interactive system that can assist the film editor in choosing and implementing arbitrarily complex shots, while taking into account some of the basic rules of cinematography. Initial experiments have shown that our tools can be a practical option for both expert and novice users, leading to an efficient and easy-to-use vertical film editing workflow.

**Acknowledgements**

This work was supported by Region Rhone Alpes (project Scenoptique) and ERC advanced grant EXPRESSIVE. All images and videos are reproduced courtesy of Célestins, Théâtre de Lyon. Special thanks to Claudia Stavisky, Auxane Dutronc and Nathalie Fillion.

## References

[Bia04] BIANCHI M.: Automatic video production of lectures using an intelligent and aware environment. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia* (New York, NY, USA, 2004), MUM '04, ACM, pp. 117–123. 2

[BLA12] BERTHOUZOZ F., LI W., AGRAWALA M.: Tools for placing cuts and transitions in interview video. *ACM Trans. Graph. 31*, 4 (July 2012), 67:1–67:8. 2

[CLM08] CHAMARET C., LE MEUR O.: Attention-based video reframing: Validation using eye-tracking. In *International Conference on Pattern Recognition* (Dec 2008), pp. 1–4. 2

[DDN08] DESELAERS T., DREUW P., NEY H.: Pan, zoom, scan: Time-coherent, trained automatic video cropping. In *Computer Vision and Pattern Recognition* (June 2008), pp. 1–8. 2

[DTS06] DALAL N., TRIGGS B., SCHMID C.: Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision* (2006). 2

[FGMR10] FELZENSZWALB P., GIRSHICK R., MCALLESTER D., RAMANAN D.: Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 32*, 9 (sept. 2010), 1627 –1645. 2

[Gan14] GANDHI V.: *Automatic Rush Generation with Application to Theatre Performances*. Phd thesis, Grenoble University, Dec. 2014. 2, 4, 5

[GKE11] GRUNDMANN M., KWATRA V., ESSA I.: Auto-directed video stabilization with robust l1 optimal camera paths. In *Computer Vision and Pattern Recognition (CVPR)* (2011). 2

[GR13] GANDHI V., RONFARD R.: Detecting and Naming Actors in Movies using Generative Appearance Models. In *Computer Vision and Pattern Recognition* (2013). 2, 4

[GRG14] GANDHI V., RONFARD R., GLEICHER M.: Multi-clip video editing from a single viewpoint. In *Computer Vision and Media Production* (2014), CVMP '14, pp. 9:1–9:10. 2, 4, 5

[GWP07] GOLDSTEIN R. B., WOODS R. L., PELI E.: Where people look when watching movies: Do all viewers look at the same place? *Comput. Biol. Med. 37*, 7 (July 2007), 957–964. 2

[HLZ04] HUA X.-S., LU L., ZHANG H.-J.: Optimization-based automated home video editing system. *IEEE Trans. Cir. and Sys. for Video Technol. 14*, 5 (May 2004), 572–583. 2

[HWG07] HECK R., WALLICK M., GLEICHER M.: Virtual videography. *ACM Trans. Multimedia Comput. Commun. Appl. 3*, 1 (Feb. 2007). 2

[JSSH14] JAIN E., SHEIKH Y., SHAMIR A., HODGINS J.: Gaze-driven video re-editing. *ACM Transactions on Graphics* (2014). 2

[KAAU02] KUMANO M., ARIKI Y., AMANO M., UEHARA K.: Video editing support system based on video grammar and content analysis. In *International Conference on Pattern Recognition* (2002), vol. 2, pp. 1031–1036 vol.2. 2

[KWK12] KAISER R., WEISS W., KIENAST G.: The fascinate production scripting engine. In *Proceedings of the 18th international conference on Advances in Multimedia Modeling* (Berlin, Heidelberg, 2012), MMM'12, Springer-Verlag, pp. 682–692. 2

[LGW*11] LIU F., GLEICHER M., WANG J., JIN H., AGARWALA A.: Subspace video stabilization. *ACM Transcations on Graphics 30*, 1 (2011), 1–10. 2

[McA09] MCAULEY G.: The video documentation of theatrical performance. *New Theatre Quarterly 10* (2009), 183–194. 2

[Mur01] MURCH W.: *In the blink of an eye*. Silman-James, 2001. 1

[RGB13] RONFARD R., GANDHI V., BOIRON L.: The Prose Storyboard Language: A Tool for Annotating and Directing Movies. In *2nd Workshop on Intelligent Cinematography and Editing part of Foundations of Digital Games - FDG 2013* (Chania, Crete, Greece, May 2013), Society for the Advancement of the Science of Digital Games. 2

[Sal06] SALT B.: *Moving into pictures*. Starword, London, 2006. 3

[SFWS13] SCHREER O., FELDMANN I., WEISSIG C.AND KAUFF P., SCHAFER R.: Ultrahigh-resolution panoramic imaging for format-agnostic video production. *Proceedings of the IEEE 101*, 1 (January 2013), 99–114. 2

[SZ03] SIVIC J., ZISSERMAN A.: Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision* (Washington, DC, USA, 2003), ICCV '03, IEEE Computer Society, pp. 1470–. 4

[TB09] THOMSON R., BOWEN C. J.: *Grammar of the shot*. Focal Press, 2009. 1, 3

[ZRCH08] ZHANG C., RUI Y., CRAWFORD J., HE L.-W.: An automated end-to-end lecture capture and broadcasting system. *ACM Trans. Multimedia Comput. Commun. Appl. 4*, 1 (Feb. 2008), 6:1–6:23. 2